DATA607 - Homework 1

Mike Dehn

9/4/2021

1 Effective and Ineffective Displays of Data

The examples shown in this submission for effective and ineffective figures compare visualizations of similar data. Example 1 in each subsection is a display of customer satisfaction data for subway systems, and Example 2 in each subsection is a display of COVID exposure source data.

1.1 Effective Displays

The following subsections provide examples of effective data visualizations.

1.1.1 Example 1

This example is taken from the San Francisco Chronicle article "BART's approval rating plummets as riders complain about filth and crime" published in January 2019[1]. The article explains how customer satisfaction in the Bay Area Rapid Transit (BART) metro system was impacted by key events and ridership. Figure 1 is used in the article to represent these trends using data collected by BART from 1996 through 2018.

This visualization is effective because it provides readers with key insights into the trends in customer satisfaction with BART's services without providing excessive ancillary data. The purpose of the chart is clear and a reader can clearly interpret the negative correlation between customer satisfaction and ridership from the plot shown. They should be cautious in drawing conclusions from these two variables alone, as there are many other factors (e.g., economics, population growth) that may contribute to these changes.

The chart also provides historical context for noteworthy events that may have had an impact on ridership and customer satisfaction. This context is provided without cluttering the plot or making the quantitative values unreadable. These provide useful information to the audience, especially since changes in public transportation may not be well-known in the general population. Although this context is useful, readers must also be wary of assuming a causal relationship between these events and changes in the data, and may need to consider whether the events were selectively included to influence opinion.

1.1.2 Example 2

This example is taken from the Maryland Department of Health's COVID contact tracing data. Health officials surveyed COVID patients throughout the pandemic, asking them to recall potential sources of COVID exposure. These responses are used to determine how the disease is acquired and transmitted throughout communities[2]. Figure 2 provides a summary level view of COVID case exposure sources, and Figure 3 provides a more detailed view of a subset of those data.

[1]Data from https://www.sfchronicle.com/bayarea/article/BART-s-approval-rating-plummets-as-riders-13550578.php
[2]Data from https://coronavirus.maryland.gov/pages/contact-tracing

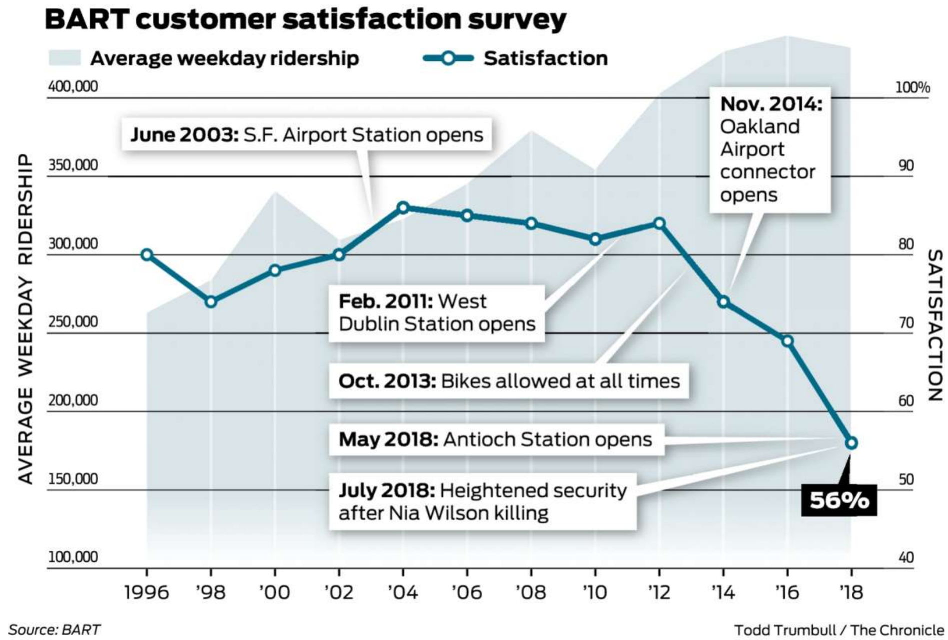


Figure 1: BART customer satisfaction versus ridership and key relevant historical events

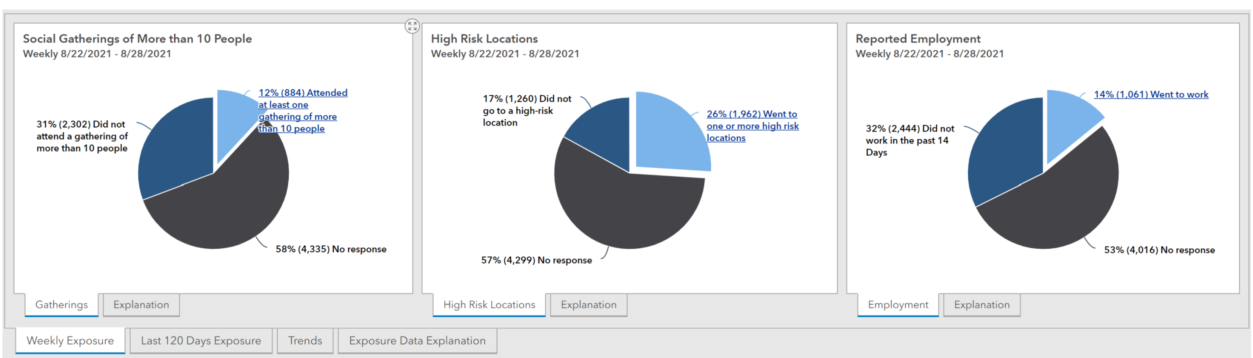


Figure 2: Summary-level COVID exposure sources based on Maryland Department of Health contact tracing

Figure 3: Detailed COVID exposure sources based on Maryland Department of Health contact tracing

Each of these figures is effective in conveying important trends and information about COVID transmission and propagation in Maryland. The summary level pie charts in Figure 2 provide high-level insight into what types of settings tend to result in COVID transmission without overwhelming the reader. The summary level view also reports how much data is missing, providing percentages of respondents who did not provide an answer. This transparency allows more refined interpretation of the contact tracing results.

If the reader is interested in further detail, bar charts like that shown in Figure 3 are available for each exposure category. Rather than providing coarse percentage values, the more specific bar charts provide specific case counts for each exposure type. Through this combination of summary and detailed data display, the Maryland Department of Health is effective in conveying important information to the reader about COVID transmission trends.

1.2 Ineffective Displays

The following subsections provide examples of ineffective data visualizations.

1.2.1 Example 1

This example is taken from the FleetLogging, Inc. study of which subway stations are the most stressful for travelers[3]. The study used a natural language processing tool, TensiStrength, to analyze stress levels in the text of Google reviews for subway stations across the globe. Researchers built reports on the relative stress levels experienced by travelers in each subway system and provided a ranking for each station. Figure 4 represents the study's findings in New York City.

The FleetLogging figure is ineffective from a graphical design perspective. The data presented are the stress percentage levels for each subway station in the NYC subway system and could be much better represented using a simple bar chart. This figure is technically a bar chart that wraps around in a circular shape. It is difficult to tell which percentage values are associated with each station since the eye must follow a circular path from the name to the percentage. A simple horizontal or vertical bar chart is much easier to understand.

The format of the figure is also misleading and might cause confusion about the study's findings. Since each bar in the chart is set at a different radius from a concentric circle, the overall area of each bar changes dramatically based on its position in the chart. The minimum stress value is 53.7 percent (Sheepshead Bay) and the maximum stress value is 66.7 percent (Jamaica Center-Parsons/Archer). Although the maximum value is approximately 24 percent larger than the minimum, its bar area is many times larger than that of the minimum. This may cause readers to misinterpret the difference in stress levels as more dramatic than they actually are.

1.2.2 Example 2

This example is taken from the Illinois Department of Public Health study on COVID-19 case exposure locations[4]. The data in this study were gathered from contact tracing of COVID-19 patients via surveys asking them to recall known exposures that may have led to their infection. Figure 5 represents the relative proportion of case exposure locations.

The COVID-19 figure is ineffective from a graphical design and informational perspective. Although a pie chart is a good choice for showing relative proportions of the values for a one-dimensional data set, the chart is unreadable because the chart is divided into over 50 sections. The percentage values are unreadable for most of the categories in data and would benefit from a less granular summary-level view of location categories.The problem is exacerbated by the original source not providing another format (e.g., a table) that contains these values. In essence, there is information reported, but lost, due to this ineffective graphic.

[3] Data from https://fleetlogging.com/most-stressful-train-stations/

[4] Data from http://www.dph.illinois.gov/covid19/location-exposure?regionID=0

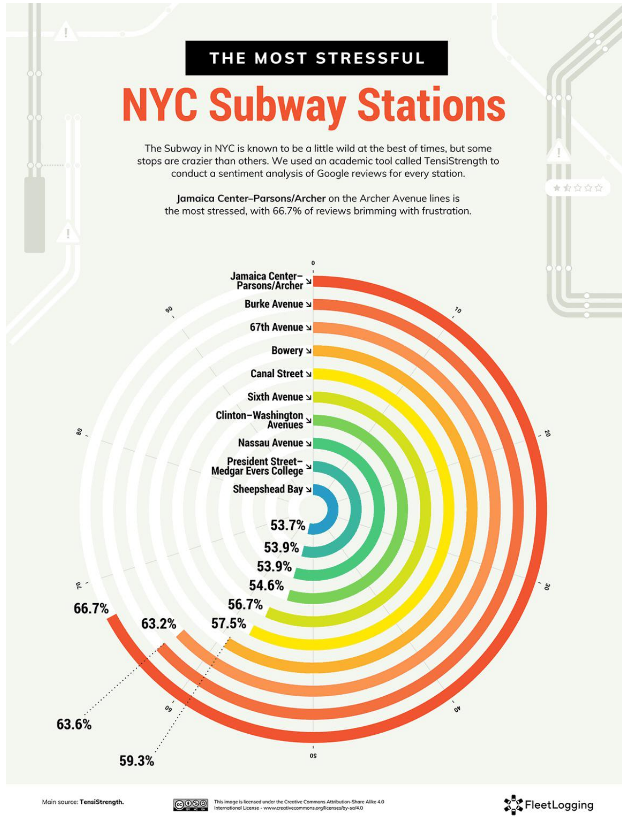


Figure 4: New York City subway station stress rankings based on TensiStrength analysis of Google reviews

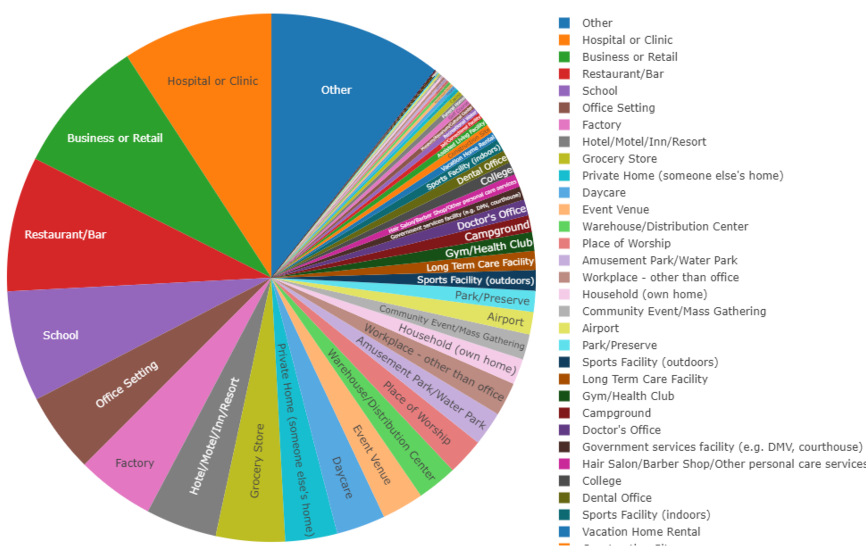


Figure 5: Prevalence of COVID case exposure sources

The chart may also be misleading since there is no “Unknown” category in the data. Some patients likely responded to the contact tracing questionnaire with an unknown exposure location, but this is not reported as a distinct category from “Other”. As a result, the audience cannot interpret the difference between unexplained exposures and those that come from a known but non-specific category.

2 Part 2

This section includes output from the original IntroExercises.R and fixes incorrect code that prevents proper execution.

The three changes that need to be made for this code to work are:

1. Comment out incorrect line `3x`, which was likely introduced as an example to show that multiplication is properly written as `3*x` in R.
2. Load tidyverse library prior to executing functions that require it (e.g., `as_tibble()`).
3. Update path to `mushroom.csv` to match that of the file system on the current machine.

The output of executing this code is shown below.

2.1 Output

```
knitr::opts_chunk$set(echo = TRUE)
# Some intro R exercises

3 + 4
```

```
## [1] 7
```

```
x<-c(1,1,2)

# WARNING! The line below produces an error. It is invalid
# syntax. Comment it out to hide from output, assuming what was
# meant is 3*x
#3x

3*x
```

```
## [1] 3 3 6
```

```
x^3
```

```
## [1] 1 1 8
```

```
t(x)%*%x
```

```
##      [,1]
## [1,]    6
```

```r
cbind(x,x)
```

```
##      x x
## [1,] 1 1
## [2,] 1 1
## [3,] 2 2
```

```r
rbind(x,x)
```

```
##   [,1] [,2] [,3]
## x    1    1    2
## x    1    1    2
```

```r
x*x
```

```
## [1] 1 1 4
```

```r
x+as.factor(x)
```

```
## Warning in Ops.factor(x, as.factor(x)): '+' not meaningful for factors
```

```
## [1] NA NA NA
```

```r
y<-matrix(runif(500^2),nrow=500)
dim(y)
```

```
## [1] 500 500
```

```r
library(Matrix)
y[y<0.8]<-0
y[y>0.8]<-1
Y<-Matrix(y)

rt<-proc.time()
C<-y%*%y%*%y
proc.time()-rt
```

```
##    user  system elapsed
##    0.16    0.00    0.15
```

```r
rt<-proc.time()
C<-Y%*%Y%*%Y
proc.time()-rt
```

```
##    user  system elapsed
##    0.09    0.00    0.10
```

```r
# WARNING! Load tidyverse before invoking functions from it
library('tidyverse')
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
```

```r
# Cars fun
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```r
is.data.frame(cars)
```

```
## [1] TRUE
```

```r
summary(cars$dist)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   26.00   36.00   42.98   56.00  120.00
```

```r
table(cars$speed)
```

```
##
##  4  7  8  9 10 11 12 13 14 15 16 17 18 19 20 22 23 24 25
##  2  2  1  1  3  2  4  4  4  3  2  3  4  3  5  1  1  4  1
```

```r
cars$speed[cars$speed<10]<-"Slow"
rm(cars)
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```
dim(cars)
```

```
## [1] 50  2
```

```
sp<-numeric(50)
sp[cars$speed<10]<-"Slow"
sp[cars$speed>=10 & cars$speed<20]<-"Medium"
sp[cars$speed>=20]<-"Fast"
cars$speed<-sp
head(cars)
```

```
##   speed dist
## 1  Slow    2
## 2  Slow   10
## 3  Slow    4
## 4  Slow   22
## 5  Slow   16
## 6  Slow   10
```

```
head(as_tibble(cars))
```

```
## # A tibble: 6 x 2
##   speed  dist
##   <chr> <dbl>
## 1 Slow      2
## 2 Slow     10
## 3 Slow      4
## 4 Slow     22
## 5 Slow     16
## 6 Slow     10
```

```
cars %>% group_by(speed) %>% summarize(Mean_dist=mean(dist))
```

```
## # A tibble: 3 x 2
##   speed  Mean_dist
##   <chr>      <dbl>
## 1 Fast        69.3
## 2 Medium      39.2
## 3 Slow        10.7
```

```
# Estimate the expected value of a truncated normal distribution
# truncated between 0 and 1
mu=1
```

```r
sigma=2
alpha=(0-mu)/sigma
beta=(1-mu)/sigma
x<-rnorm(10000,mu,sigma)
w<-(x>0)&(x<1)
ind<-which(w==TRUE)
mean(x[ind])
```

```
## [1] 0.512496
```

```r
# true value
mu+(dnorm(alpha)-dnorm(beta))/(pnorm(beta)-pnorm(alpha))*sigma
```

```
## [1] 0.5103275
```

```r
# compute the sum of n^p

int.power<-function(n,p){
  # n is nonnegative integers
  # p is a (possibly negative) integer
  # compute the sum of x^p for x=1,2,...,n
  return(sum(c(1:n)^p))
}

sapply(1:10,function(x) int.power(x,1))
```

```
##  [1]  1  3  6 10 15 21 28 36 45 55
```

```r
sapply(1:10,function(x) int.power(x,2))
```

```
##  [1]   1   5  14  30  55  91 140 204 285 385
```

```r
sqrt(int.power(100,-2)*6)
```

```
## [1] 3.132077
```

```r
sqrt(int.power(1000,-2)*6)
```

```
## [1] 3.140638
```

```r
sqrt(int.power(10000,-2)*6)
```

```
## [1] 3.141497
```

```r
int.power(5.4444,2.4)
```

```
## [1] 95.69361
```

```r
int.power(5,2.4)
```

```
## [1] 95.69361
```

```r
1:5.4444
```

```
## [1] 1 2 3 4 5
```

```r
library(gapminder)
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

```r
table(gapminder$year)
```

```
## 
## 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007 
##  142  142  142  142  142  142  142  142  142  142  142  142
```

```r
gap2007<-gapminder %>% filter(year==2007)
head(gap2007)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       2007    43.8 31889923      975.
## 2 Albania     Europe     2007    76.4  3600523     5937.
## 3 Algeria     Africa     2007    72.3 33333216     6223.
## 4 Angola      Africa     2007    42.7 12420476     4797.
## 5 Argentina   Americas   2007    75.3 40301927    12779.
## 6 Australia   Oceania    2007    81.2 20434176    34435.
```

```r
gap2007 %>% group_by(continent) %>% summarize(lifeExp=median(lifeExp))
```

```
## # A tibble: 5 x 2
##   continent lifeExp
##   <fct>       <dbl>
## 1 Africa       52.9
## 2 Americas     72.9
## 3 Asia         72.4
## 4 Europe       78.6
## 5 Oceania      80.7
```

```r
library(socviz)
dim(gss_sm)
```

```
## [1] 2867   32
```

```r
gss<-na.omit(gss_sm)
dim(gss)
```

```
## [1] 668  32
```

```r
names(gss)
```

```
##  [1] "year"        "id"          "ballot"      "age"         "childs"
##  [6] "sibs"        "degree"      "race"        "sex"         "region"
## [11] "income16"    "relig"       "marital"     "padeg"       "madeg"
## [16] "partyid"     "polviews"    "happy"       "partners"    "grass"
## [21] "zodiac"      "pres12"      "wtssall"     "income_rc"   "agegrp"
## [26] "ageq"        "siblings"    "kids"        "religion"    "bigregion"
## [31] "partners_rc" "obama"
```

```r
gss %>% group_by(bigregion, religion) %>% summarize(n=length(id)) %>%
  ungroup %>% group_by(bigregion) %>%
  mutate(proportion = n / sum(n))
```

```
## `summarise()` has grouped output by 'bigregion'. You can override using the `.groups` argument.
```

```
## # A tibble: 20 x 4
## # Groups:   bigregion [4]
##    bigregion religion       n proportion
##    <fct>     <fct>      <int>      <dbl>
##  1 Northeast Protestant    40    0.310
##  2 Northeast Catholic      46    0.357
##  3 Northeast Jewish        10    0.0775
##  4 Northeast None          28    0.217
##  5 Northeast Other          5    0.0388
##  6 Midwest   Protestant    86    0.506
##  7 Midwest   Catholic      52    0.306
##  8 Midwest   Jewish         1    0.00588
##  9 Midwest   None          27    0.159
## 10 Midwest   Other          4    0.0235
## 11 South     Protestant   138    0.639
## 12 South     Catholic      36    0.167
## 13 South     Jewish         2    0.00926
## 14 South     None          32    0.148
## 15 South     Other          8    0.0370
## 16 West      Protestant    66    0.431
## 17 West      Catholic      26    0.170
## 18 West      Jewish         5    0.0327
## 19 West      None          41    0.268
## 20 West      Other         15    0.0980
```

```r
# Fun with mushrooms
# WARNING! Update path to mushroom.csv file for this machine
mushroom <- read.csv("~/R/data607/hw1/mushroom.csv", header=FALSE)
mushroom <- as_tibble(mushroom)
head(mushroom)
```

```
## # A tibble: 6 x 23
##   V1    V2    V3    V4    V5    V6    V7    V8    V9    V10   V11   V12   V13
##   <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 p     x     s     n     t     p     f     c     n     k     e     e     s
## 2 e     x     s     y     t     a     f     c     b     k     e     c     s
## 3 e     b     s     w     t     l     f     c     b     n     e     c     s
## 4 p     x     y     w     t     p     f     c     n     n     e     e     s
## 5 e     x     s     g     f     n     f     w     b     k     t     e     s
## 6 e     x     y     y     t     a     f     c     b     n     e     c     s
## # ... with 10 more variables: V14 <chr>, V15 <chr>, V16 <chr>, V17 <chr>,
## #   V18 <chr>, V19 <chr>, V20 <chr>, V21 <chr>, V22 <chr>, V23 <chr>
```

```r
mushroom <- mushroom %>% mutate_if(is.character,as.factor)
head(mushroom)
```

```
## # A tibble: 6 x 23
##   V1    V2    V3    V4    V5    V6    V7    V8    V9    V10   V11   V12   V13
##   <fct> <fct> <fct> <fct> <fct> <fct> <fct> <fct> <fct> <fct> <fct> <fct> <fct>
## 1 p     x     s     n     t     p     f     c     n     k     e     e     s
## 2 e     x     s     y     t     a     f     c     b     k     e     c     s
## 3 e     b     s     w     t     l     f     c     b     n     e     c     s
## 4 p     x     y     w     t     p     f     c     n     n     e     e     s
## 5 e     x     s     g     f     n     f     w     b     k     t     e     s
## 6 e     x     y     y     t     a     f     c     b     n     e     c     s
## # ... with 10 more variables: V14 <fct>, V15 <fct>, V16 <fct>, V17 <fct>,
## #   V18 <fct>, V19 <fct>, V20 <fct>, V21 <fct>, V22 <fct>, V23 <fct>
```

```r
colnames(mushroom) <- c("edibility", "cap_shape", "cap_surface",
                        "cap-color", "bruises", "odor",
                        "gill-attachement", "gill-spacing", "gill-size",
                        "gill-color", "stalk-shape", "stalk-root",
                        "stalk-surface-above-ring", "stalk-surface-below-ring",
                        "stalk-color-above-ring",
                        "stalk-color-below-ring", "veil-type", "veil-color",
                        "ring-number", "ring-type", "spore-print-color",
                        "population", "habitat")

head(mushroom)
```

```
## # A tibble: 6 x 23
##   edibility cap_shape cap_surface `cap-color` bruises odor  `gill-attachement`
##   <fct>     <fct>     <fct>       <fct>       <fct>   <fct> <fct>
## 1 p         x         s           n           t       p     f
## 2 e         x         s           y           t       a     f
## 3 e         b         s           w           t       l     f
## 4 p         x         y           w           t       p     f
```

```
## 5 e         x         s           g           f       n     f
## 6 e         x         y           y           t       a     f
## # ... with 16 more variables: gill-spacing <fct>, gill-size <fct>,
## #   gill-color <fct>, stalk-shape <fct>, stalk-root <fct>,
## #   stalk-surface-above-ring <fct>, stalk-surface-below-ring <fct>,
## #   stalk-color-above-ring <fct>, stalk-color-below-ring <fct>,
## #   veil-type <fct>, veil-color <fct>, ring-number <fct>, ring-type <fct>,
## #   spore-print-color <fct>, population <fct>, habitat <fct>
```

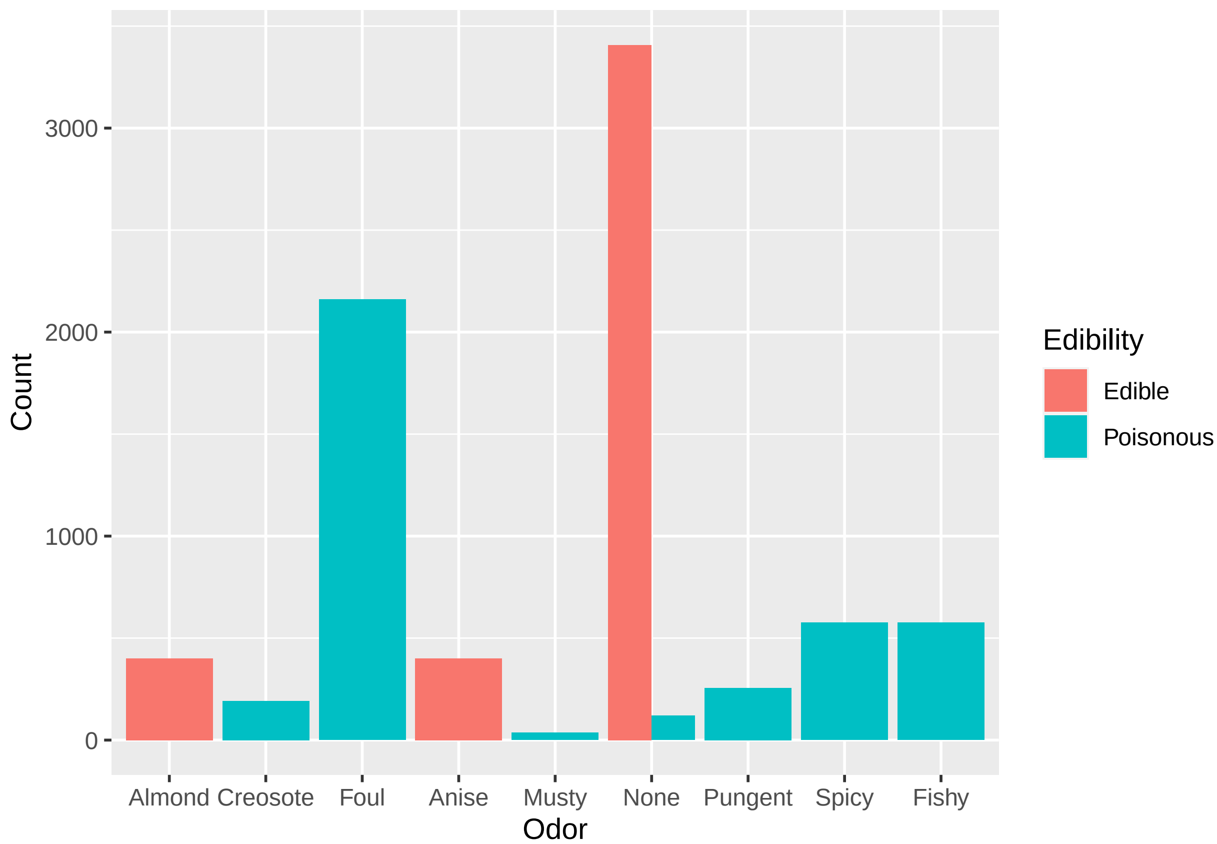
```
table(mushroom$edibility)
```

```
##
##    e    p
## 4208 3916
```

```
table(mushroom$edibility,mushroom$odor)
```

```
##
##        a    c    f    l    m    n    p    s    y
##   e  400    0    0  400    0 3408    0    0    0
##   p    0  192 2160    0   36  120  256  576  576
```

```
ggplot(data=mushroom, aes(fill=edibility,x=odor)) + geom_bar(position="dodge")+
 scale_fill_discrete(name = "Edibility", labels = c("Edible", "Poisonous"))+
  scale_x_discrete(labels=c("Almond", "Creosote", "Foul", "Anise", "Musty", "None",
                            "Pungent", "Spicy", "Fishy"))+
  xlab("Odor") +
  ylab("Count")
```



3000
2000
1000
0
Count
Almond
Creosote
Foul
Anise
Musty
None
Pungent
Spicy
Fishy
Odor
Edibility
Edible
Poisonous